Confidences for Commonsense Reasoning

Tanel Tammet, Dirk Draheim, Priit Järv

Tallinn University of Technology, Tallinn, Estonia

Contents

- Motivation
- Context
- Sources, representation and meaning
- Confer framework
- Implementation: comparing calculation results and performance
- Work in progress

Motivation

An icy tale featuring A.R., M.L. and A.I.

Automated reasoning (A.R.) used to be a major part of A.I. research.

An icy tale featuring A.R., M.L. and A.I.

Automated reasoning (A.R.) used to be a major part of A.I. research.

A.I. winters, bad rap of Cyc, sem web failures etc reduced interest in A.R.

An icy tale featuring A.R., M.L. and A.I.

Automated reasoning (A.R.) used to be a major part of A.I. research.

A.I. winters, bad rap of Cyc, sem web failures etc reduced interest in A.R.

A.I was taken over by machine learning (M.L.)

An icy tale featuring A.R., M.L. and A.I.

Automated reasoning (A.R.) used to be a major part of A.I. research.

A.I. winters, bad rap of Cyc, sem web failures etc reduced interest in A.R.

A.I was taken over by machine learning (M.L.)

A.R. lived in the cold dark caves of verification, interactive math, etc.

An icy tale featuring A.R., M.L. and A.I.

Automated reasoning (A.R.) used to be a major part of A.I. research.

A.I. winters, bad rap of Cyc, sem web failures etc reduced interest in A.R.

A.I was taken over by machine learning (M.L.)

A.R. lived in the dark niches of verification, interactive math, etc.

M.L., though mighty, turned out to be somewhat dumb.

An icy tale featuring A.R., M.L. and A.I.

Automated reasoning (A.R.) used to be a major part of A.I. research.

A.I. winters, bad rap of Cyc, sem web failures etc reduced interest in A.R.

A.I was taken over by machine learning (M.L.)

A.R. lived in the dark niches of verification, interactive math, etc.

M.L., though mighty, turned out to be somewhat dumb.

People started hoping for a powerful hybrid: M.L. / A.R. combo.

An icy tale featuring A.R., M.L. and A.I.

Automated reasoning (A.R.) used to be a major part of A.I. research.

A.I. winters, bad rap of Cyc, sem web failures etc reduced interest in A.R.

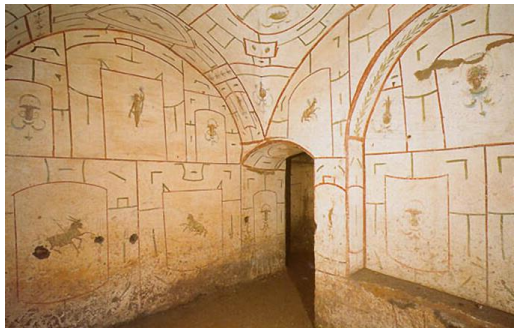
A.I was taken over by machine learning (M.L.)

A.R. lived in the dark niches of verification, interactive math, etc.

M.L., though mighty, turned out to be somewhat dumb.

People started hoping for a powerful hybrid: M.L. / A.R. combo.

But A.R. was still hidden in the safe catacombs: nowhere to be seen.



An icy tale featuring A.R., M.L. and A.I.

Automated reasoning (A.R.) used to be a major part of A.I. research.

A.I. winters, bad rap of Cyc, etc reduced interest in A.R.

A.I was taken over by machine learning (M.L.)

A.R. lived in the dark niches of verification, interactive math, etc.

M.L., though mighty, turned out to be somewhat dumb.

People started hoping for a powerful hybrid: M.L. / A.R. combo.

But A.R. was still hidden in the safe catacombs: nowhere to be seen.

Bring A.R. back to the sunny fields of A.I. to work alongside M.L!

A.R. needs a wealth of capabilities for commonsense A.I.:

- Find answers, i.e. substitutions
- Survive contradictions
- Estimate confidences
- Handle defeasible reasoning
- Handle analogy-based reasoning
- Manage context
- Integrate with NLP systems
- Integrate with knowledge bases

Why not enhance the existing A.R. systems towards these goals?

Context

Practical context of the paper: NLP Q&A / dialogue

- Suppose we use an A.R. component as a part of an NLP Q&A / dialogue system.
- Most facts and rules are uncertain: some more, some less.
- We need to estimate our confidence in the results of the derivations.

Using real probabilities does not seem to be realistic for this scenario.

Probabilities and the like: big picture

Philosophy and math:

- https://plato.stanford.edu gives six different “main interpretations”
- Frequentism: full information, combinatorics, Kolmogorov axioms
- Bayesianism: branch into objectivism and subjectivism

Probabilities + logic in computer science:

- Multi-valued logics
- Fuzzy logic: degrees of properties, not belief; no cumulating evidence
- Probabilistic logics and systems: next slide

Current approaches outside fuzzy logic

- Programming tools for real probabilities with distributions, etc
- Propositional logic + probabilities: Bayesian networks
- Logic programming with probabilities: grounding a la Problog
- Markov logic networks, also grounding

Systems for the last two cannot handle

```
1.0 ::  p(a).
1.0 ::  p(i(a,b)).
1.0 ::  p(Y) :- p(X), p(i(X,Y)).
query( p(b) ).
```

Our approach to sources, representation and meaning

Sources of confidence

We follow the subjective interpretation of probability as a degree of belief, originating from Ramsey and De Finetti

We assume that the confidence in a fact or rule in our common sense knowledge base (KB in the following) typically arises from

- a large number of human users via crowd-sourcing like ConceptNet
- NLP-analyzed scraped text from the web like NELL
- combining different knowledge bases with weights like Quasimodo
- assumptions accompanying a question

Assigning confidences to statements

- To each FOL statement S we will assign both a confidence c and a set L of unique identiers of (non-derived) input statements used for deriving this statement: a triple <S, c, L>.
- Each S in the triple is a clause.
- Lists of such triples are then treated as sets.
- The dependency lists L are used in the formula estimating the cumulated condence.
- NB! For each single FOL clause S there may be many different derivable triples

  <S, c , L> for different c and L, stemming from different derivation trees of S.

Interpretation of confidences

- Confidence estimates the lower limit of the probability of a statement:

  <S, c, L> means “statements L support the claim that probability(S) ≥ c".

- An example, from which we derive bird(a) : 0:682

  <bird(X), 0.1, L1>
  <bird(a), 0.8, L2>
  <bird(a), 0.9, L3>
  <not bird(a), 0.3, L4>

Confer framework

CONFER framework: preliminaries

- Extends the resolution method: underlying FOL kept intact.
- The exact maximal confidence for derived statements is not calculated.
- Assume the question posed is in one of two forms:

  (1) Is the statement Q true?

  (2) Find values V for existentially bound variables in Q so that Q is true.

$$exist\ X_1, \ldots, X_n\ \ (\mathrm{Q}(\mathrm{X}_1, \ldots, \mathrm{X}_\mathrm{n})\ \ \&\ \ \mathrm{not\ answer}(\mathrm{X}_1, \ldots, \mathrm{X}_\mathrm{n}))$$

- A clause consisting of only answer predicates is called an “answer clause”

Confer framework: main points

- Contradictions not containing an answer clause are discarded.
- The proof search does not stop whenever an answer clause is found, but will continue to look for new answer clauses until a predetermined time limit is reached.
- Resolution / paramodulation rules decrease the confidence of the result by multiplying of the confidences of arguments.
- Different independent derivations of the same answer clause are combined to cumulatively increase the confidence of the result.
- Attempt to find proofs for negations of the instantiated query answers.
- The difference of confidences of positive / negative query are final numbers.

Time limits are crucial

Let *t* be total time given.

Spend t / 2 time for finding positive answers, combined to *N* answers.

Spend *(t / 2) / N* time for searching proofs for negations of answers.

Optimistic approach: more time for positive search.

Pessimistic approach: more time for negative search.

Algorithm 1 CONFER algorithm

Input: Common sense knowledge base KB, question Q, time limit t.
Output: Set of answers R with attached confidences.

1: Let R={}.
2: Find a set of initial positive answers with confidences and dependencies IPA=$\{\langle A_1, c_1, L_1\rangle, ..., \langle A_p, c_p, L_p\rangle\}$ for Q from KB using c-resolution with the time limit $t/2$.
3: Calculate a set of cumulative positive answers CPA=$\{\langle B_1, d_1, E_1\rangle, ..., \langle B_r, d_r, E_r\rangle\}$ from IPA.
4: Let $i = 1$.
5: **while** $i <= r$ **do**
6: Form the negated question NQ_i from $\neg Q$ with a substition s given by B_i
7: Find a set of initial negative answers N_i with confidences and dependencies for NQ_i from KB using c-resolution with the time limit $t/(2 * r)$.
8: **if** N_i is empty **then**
9: Let $nc_i = 0$.
10: **else**
11: Calculate the cumulative negative confidence nc_i from N_i.
12: **end if**
13: Add a pair $\langle B_i, (c_i - nc_i)\rangle$ to R.
14: Let $i = i + 1$.
15: **end while**
16: For each pair $\langle B_i, d_i\rangle$, $\langle B_j, d_j\rangle$ in R where $i \neq j$, $B_i = B_j$ and $d_i >= d_j$, remove $B_j : d_j$ from R.
17: Remove from R all elements $\langle B_i, d_i\rangle$ where $d_i <= 0$.
18: **return** the set of answers with confidences R.

Basic confidence calculation for resolution steps

Resolution and paramodulation: multiply the confidences of the premises.
Factorization: unchanged. Question clauses have a confidence 1.

Example: from

```
0.8:: bird(tweety).
0.9:: bird(X) => canfly(X).
0.7:: canfly(X) => fast(X).
1.0:: fast(X) => answer(X).
```

we derive

```
0.72:: canfly(tweety).
0.504:: fast(tweety).
0.504:: answer(tweety).
```

Notes

We can give a higher confidence in special cases: multiplication is the lower limit.

Subsumption / demodulation have to be modified to account for the confidence:

A triple T1 = <A1, c1, L1> consisting of a clause A1, confidence c1 and a dependency list L1 c-subsumes a triple T2 = <A2, c2, L2> if and only if

- A1 subsumes A2,
- c1 ≥ c2 and
- L1 is a (non-strict) subset of L2.

Cumulative confidence: independence

A clause C can be derived in different ways, giving two different derivations d1 and d2 with confidences c1 and c2.

In case the derivations d1 and d2 are independent, we could apply the standard formula P(A or B) = P(A) + P(B) - P(A and B)

We will estimate the independence *i* of two derivations d1 and d2 simply as

$$1 - \frac{number\ of\ shared\ input\ clauses\ of\ d_1 and\ d_2}{total\ number\ of\ input\ clauses\ in\ d_1 and\ d_2}$$

Thus, if no clauses are shared between d1 and d2, then i = 1
and if all the clauses are shared, then i = 0.

Cumulative confidence calculation

Given two derivations d1 and d2 of the search result C with confidences c1 and c2, calculate the updated confidence of C as

$$max(c1 + c2*i*h,\ \ c1*i*h + c2)\ -\ c1*c2*i*h$$

where

- independence of derivations *i* is defined as above,
- *h* is the heuristic estimate of the independence of the total set of input clauses from 1 for total independence to 0 for total dependence.

Cumulative confidence calculation: intuition

If d1 and d2 do not share non-question input clauses and all the input clauses are mutually independent, $i*h = 1$ and the formula turns into

$$c1+c2-(c1*c2).$$

If d1 and d2 have the same non-question input clauses or the total set of input clauses is mutually totally dependent, $i*h = 0$ and the formula turns into

$$\max(c1, c2).$$

Cumulative confidence: example

Example: from

```
0.8:: bird(tweety).
0.9:: bird(X) => canfly(X).
0.7:: canfly(X) => fast(X).
0.8:: inair(tweety).
0.9:: inair(X) => fast(X).
1.0:: fast(X) => answer(X).
```

we derive

```
0.72:: canfly(tweety).
0.504:: fast(tweety).
0.504:: answer(tweety).
```

Derivation d1

```
0.72:: fast(tweety).
0.72:: answer(tweety).
```

Derivation d2

Derivations d1 and d1 do not share clauses, thus i=1.
Let heuristic be 1 as well.
Then we get cumulative confidence
$0.504 + 0.72 - (0.504 * 0.72) = 0.86112$.

Now let heuristic h=0.5 (partial independence).
Then we get cumulative confidence
$\max(0.504 + 0.72 * 0.5 - (0.504 * 0.72 * 0.5),$
$0.504 * 0.5 + 0.72 - (0.504 * 0.72 * 0.5)) =$
$\max(0.68256, 0.79056) =$
0.79056

Setting h to 0 (total dependence) leads to cumulative confidence
$\max(0.504, 0.72) =$
0.72

Negation: initial issues

We attempt to find both positive and negative evidence for any potential answer.
This cannot be easily done in a single proof search run.

Giving a general search question containing variables like

```
bird(X) | answer(X)
```

may produce a different set of answers than the positive question :

```
~bird(X) | answer(X)
```

The potential set of answers may be huge for both positive and negative answers.

Negation vs tiny confidence numbers

In our framework this does not mean `bird(tweety)` is unlikely to hold:

```
0.001::  bird(tweety).
```

It means that there is very little evidence for the statement.

To state that `bird(tweety)` is unlikely, state the negation explicitly:

```
0.999::  not bird(tweety).
```

Observe that both a positive and a negative statement can live happily together and they are – typically - statements from different sources.

Note: the systems we use for comparison

ProbLog2

- A leading system for probabilistic logic programs
- Live web site https://dtai.cs.kuleuven.be/problog/

Alchemy 2

- A leading system for Markov Logic
- Versions from https://github.com/PhDP/alchemy2
- Three different algorithms: MC-SAT, exact and approximate probabilistic theorem proving.

Negation examples: expected and unexpected

Consider

```
0.5 :: bird(a).
0.5 :: not bird(a).
query( bird(a) ).
```

- Confer gives confidence 0, which we interpret as “no information”, not as “false".
- The Problog2 system gives 0.25, explained by one of the authors as “an atom (head) is satisfied if any of the rules that make it true fire and none of the rules that make it false fire, hence 0.5 * (1 – 0.5) = 0.25”
- The three different algorithms of the Alchemy 2 system give answers 0.015, 0 and 0.082, respectively.

Negation: consistencies and inconsistencies

Consider

```
0.5::bird(a).
0.5::not bird(a).
0.9:: flies(X) :- bird(X).
query(flies(_)).
```

- Confer gives us 0.45: inconsistent with the result of the previous example.
- ProbLog2 gives 0.225: unintuitive, but consistent with the unintuitive result of ProbLog2 in the previous example.
- The three algorithms of Alchemy 2 give us 0.047, 0 and 0.98.

Implementation:
comparing calculation results and performance

Implementation

Confer is implemented in C as an extended version of our high-performance open-source automated reasoning system gkc https://github.com/tammet/gkc for FOL. It uses our shared memory database http://whitedb.org/ for core data structures.

The compiled executable can be downloaded from

http://logictools.org/confer/

along with a number of examples.

Social network

```
0.8::stress(ann).
0.4::stress(bob).
0.6::influences(ann,bob).
0.2::influences(bob,carl).
smokes(X) :- stress(X).
smokes(X) :- influences(Y,X), smokes(Y).
query(smokes(carl)).
```

- Confer: 0.1201, cumulating values 0.096 and 0.08
- Problog2: 0.1376
- Alchemy2 algorithms: 0.135, 0 and 0.741

Earthquake

```
person(john).
person(mary).
0.7::burglary.
0.2::earthquake.
0.9::alarm :- burglary, earthquake.
0.8::alarm :- burglary, \+earthquake.
0.1::alarm :- \+burglary, earthquake.
0.8::calls(X) :- alarm, person(X).
0.1::calls(X) :- \+alarm, person(X).
evidence(calls(john),true).
evidence(calls(mary),true).
query(burglary).
query(earthquake).
```

query	CONFER	CONFER +	CONFER -	Problog	Alch i	Alch e	Alch a
burglary	0.8713	0.97650	0.1051	0.9819	0.709	0	0.905095
earthquake	0.1648	0.8854	0.7206	0.2268	0.204	0	0.888

Default penguin example with confidences

```
1.0::bird(tweety).
1.0::penguin(pennie).
1.0:: bird(X) :- penguin(X).
0.001:: penguin(X) :- bird(X).
0.9:: flies(X) :- bird(X).
1.0:: not flies(X) :- penguin(X).
query(flies(_)).
```

query	CONFER	CONFER +	CONFER -	Problog	Alch i	Alch e	Alch a
flies(pennie)	-0.1	0.9	1.0	0	0.00001	0	0
flies(tweety)	0.899	0.9	0.001	0.8991	0.064	0	0.873

Performance

Problem	CONFER	CONFER 1.0	gkc pure	gkc full
Steamroller	0.0018	0.0015	0.001	0.06
Dreadbury	0.0017	0.0011	0.001	0.06
Lukasz 0		0.0916	0.093	0.22
Lukasz 1	0.913			
Lukasz 2	23			
Lukasz 3	19			
Lukasz 4	16			
CSR025+5	0.0004	0.0001	0.0001	4.5
CSR035+5	0.0001	0.0001	0.07	4.6
CSR045+5	3.418	1.4	1.3	5.8
CSR055+5	0.0001	0.0001	0.0001	4.5

Work in progress

Towards the A.R. component for NLP Q&A

- Implemented an alpha version of first-order defeasible reasoning with inheritance hierarchies on top of the confer system.
- Built an early version of the knowledge base with confidences and default rules with priorities, using the Quasimodo knowledge base from Max Planck for confidence-enhanced triplets and the Wordnet data for the taxonomy hierarchy used for adding default rules to Quasimodo.
- Figuring out the best ways to handle
  (a) the inconsistencies arising from the negation,
  (b) reasoning by analogy (“queen is like a king, but female”),
  (c) context.
- Selecting / combining semantic parsers to build a suitable NLP interface.

Approaching defeasible reasoning in FOL

- Do not try to check the rule blockers during the standard proof search.
- Use a special `blocker(…)` predicate similar to the conventional `answer(…)` to collect the substitutions and default rule blockers to be checked.
- Check the blockers found in the proofs, giving each check – a separate proof search - a fraction of the total time.
- Use the taxonomy hierarchies of blockers to prohibit using more general ones in the blocker check.
- Check the blockers of the successful blocker checking search in an iteratively deepening manner with smaller and smaller time limits.